Algorithmic Aspects of Telecommunication Network
TE/CS6383

Project 3
Network Reliability

Submitted by:
Shweta Ingale
Net id:ssi190000

Content

Introduction:

The aim of the project is to study experimentally how the network reliability depends on the individual link reliabilities, in the specific situation.

Network topology: A complete undirected graph on n = 5 nodes. This means, every node is connected with every other one (parallel edges and selfloops are excluded in this graph). As a result, this graph has m = 10 edges, representing the links of the network.

Components that may fail: The links of the network may fail, the nodes are always up. The reliability of each link is p, the same for every link. The parameter p will take different values in the experiments.

Reliability configuration: The system is considered operational, if the network is connected.

Problem Description:

Assumptions on the reliability model:
• Each component has two possible states: operational or failed.
• The failure of each component is an independent event.
• Component i is functioning (operational) with probability pi and is inoperational (failed) with probability 1 − pi. (These probabilities are usually known.)
• The reliability R of the system is some function of the component reliabilities:
R = f(p1,p2,...,pN) where N is the number of components.
The function f(...) above depends on the configuration, which defines when the system is considered operational, given the states of the components.

Requirement:

- Run the program for different values of p. Let the parameter p run over the [0,1] interval, in steps of 0.05. Show graphically in a diagram how the obtained network reliability values depend on p.
- Fix the p parameter at p = 0.85, and do the following experiment. Among the 210 = 1024 possible combinations of component states pick k of the combinations randomly, and flip the corresponding system condition. That is, if the system was up, change it to down, if it was down, change it to up. This aims at modeling the situation when there is some random error in the system status evaluation. Show in a diagram, how the reliability of the system changes due to this alteration, by showing how the change depends on k, in the range k = 0,1,2,3,...,20. During this 3 | P a g e experiment keep the value of the parameter p fixed at p = 0.85. To reduce the effect of randomness, run several experiments and average them out, for each value of k.

Goals:

• Debugging: It is possible by using print statements at various points. Moreover, null values and other exceptions have also been taken care of.
• Correctness: Adjacency matrix is used for checking the correctness and from the produced outputs we can see that probability and reliability values are between 0 and 1 and that is what was desired.
• Changes: They are easy to incorporate as we have used a modular approach so one change in a function can impact the whole program without requiring to make multiple changes.

NETWORK TOPOLOGY

We simulate the network topology as an undirected graph with 5 nodes and 10 edges. In this topology, every node is connected to every other node, and we avoid parallel and self loops, making the number of edges in the network to be 10. This leads to 2^10 = 1024 possible combinations in which the network topology states.

LINK RELIABILITY

The link reliability or link probability is p. For the first part of the experiment, we vary the reliability p between [0,1] in steps of 0.05. For each of these values of p, we will find the network reliability. For the second part of the experiment, we flip the states of links from k randomly chosen network combinations out of the 1024 combinations. Here, k goes from 0 to 20 and for each of these k values, we find the network reliability.

Pseudo Code:

Begin
a) Reliability for the value of p. p can take values between 0.05 and 1 which is incremented in steps of 0.05.
b) adjacencyMatrix [1024][10] is equal to the set of all combinations of 0's and 1's for 1024 combinations.

INPUT:
=> Network topology: an undirected complete graph G with n nodes,
[total edges = (n * (n -1)) / 2] => Link probability p : probability that link is up

c) Probability value for each row having 'UP' state is calculated and it is then added to the rowProb variable. For each p value of sum is calculated.
d) Reliability for different values of 'k'(0 to 20). P is set as 0.9.
e) For (int i=0 to i=k), we consider a random array ranGenArray[] which stores random numbers according to the values of k.
f) We consider a build New newAdjacencyMatrix [] which indicates the flipped values of 0's and 1's for the values of array ranGenArray[].
g) Repeat step 3 and calculate sum again.
h) Print the value of the sum for every value of 'k' from 0 to 20.

OUTPUT:
Reliability of a network, R

- Generate all the states of the graph
- Set reliability = 1
- For each state that belong to all states (every combination) do if G is connected then find the reliability r for each state.
- Calculate and return R by summing up all r (by adding the reliabilities of each network state)

End

Implementation Details:

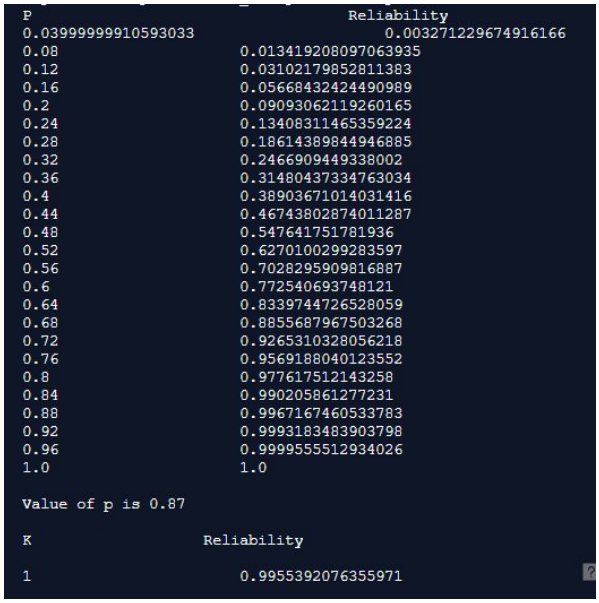
Technologies used:

Programming Language :: Java
Operating System :: Windows 10
Development Environment :: Eclipse Neon

ReadMe Section:

Create one java class with a name Project3 in Eclipse.

- Copy the code from ATN_Project_3.pdf file into java class file Project3.
- Run the Project3 java class.
- Output can be seen from console.(output shown below are from repl.it)

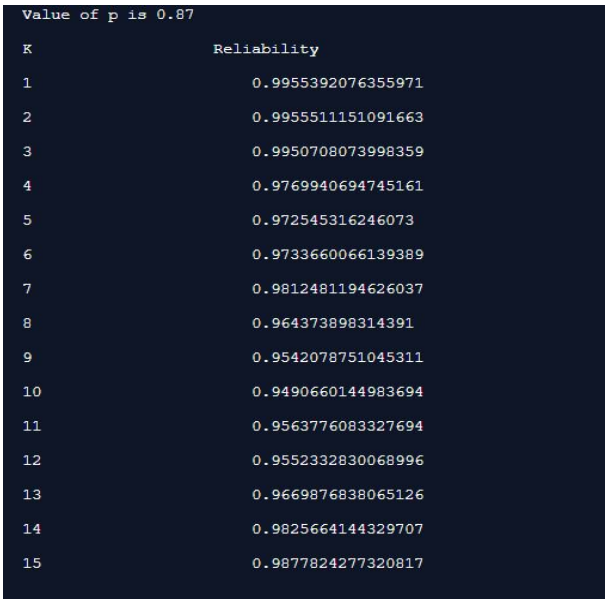


```
P                                   Reliability
0.03999999910593033                 0.003271229674916166
0.08                    0.013419208097063935
0.12                    0.03102179852811383
0.16                    0.05668432424490989
0.2                     0.09093062119260165
0.24                    0.13408311465359224
0.28                    0.18614389844946885
0.32                    0.2466909449338002
0.36                    0.31480437334763034
0.4                     0.38903671014031416
0.44                    0.46743802874011287
0.48                    0.547641751781936
0.52                    0.6270100299283597
0.56                    0.7028295909816887
0.6                     0.772540693748121
0.64                    0.8339744726528059
0.68                    0.8855687967503268
0.72                    0.9265310328056218
0.76                    0.9569188040123552
0.8                     0.977617512143258
0.84                    0.990205861277231
0.88                    0.9967167460533783
0.92                    0.9993183483903798
0.96                    0.9999555512934026
1.0                     1.0

Value of p is 0.87

K                   Reliability

1                       0.9955392076355971
```



```
Value of p is 0.87

K                   Reliability

1                       0.9955392076355971

2                       0.9955511151091663

3                       0.9950708073998359

4                       0.9769940694745161

5                       0.972545316246073

6                       0.9733660066139389

7                       0.9812481194626037

8                       0.964373898314391

9                       0.9542078751045311

10                      0.9490660144983694

11                      0.9563776083327694

12                      0.9552332830068996

13                      0.9669876838065126

14                      0.9825664144329707

15                      0.9877824277320817
```

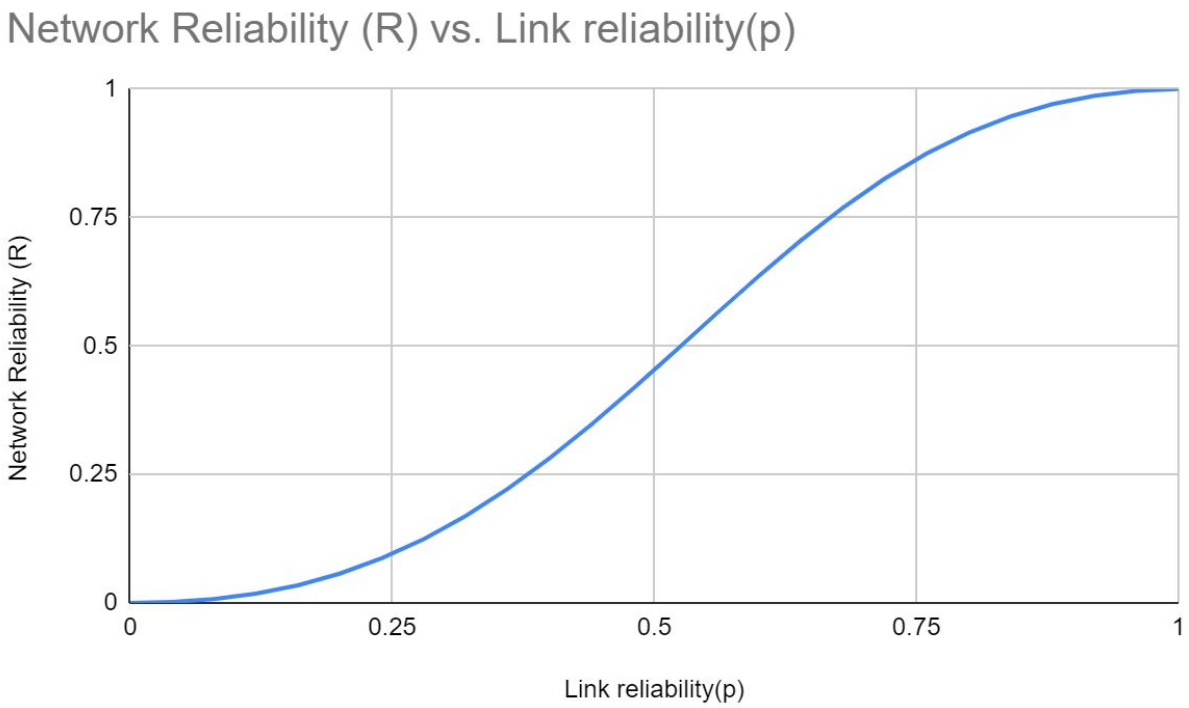
Experimental Values:

Link reliability(p)	Network Reliability (R)
0	0
0.04	0.0017
0.08	0.0074
0.12	0.018
0.16	0.0342
0.2	0.0569
0.24	0.0867
0.28	0.124
0.32	0.1691
0.36	0.2217
0.4	0.2814
0.44	0.3471
0.48	0.4173
0.52	0.4903
0.56	0.5641
0.6	0.6365
0.64	0.7055
0.68	0.7691
0.72	0.8259
0.76	0.8748
0.8	0.9152
0.84	0.9471
0.88	0.9709
0.92	0.9872
0.96	0.9968
1	1

k	Average Network Reliability
1	0.96569
2	0.96581
3	0.96632
4	0.966
5	0.96766
6	0.94939
7	0.93143
8	0.93873
9	0.94241
10	0.924888
11	0.92828
12	0.90495
13	0.93417
14	0.93927
15	0.93233
16	0.95631
17	0.9456
18	0.94964
19	0.951504
20	0.94799
21	0.95864
22	0.96305
23	0.92512
24	0.93806
25	0.95121

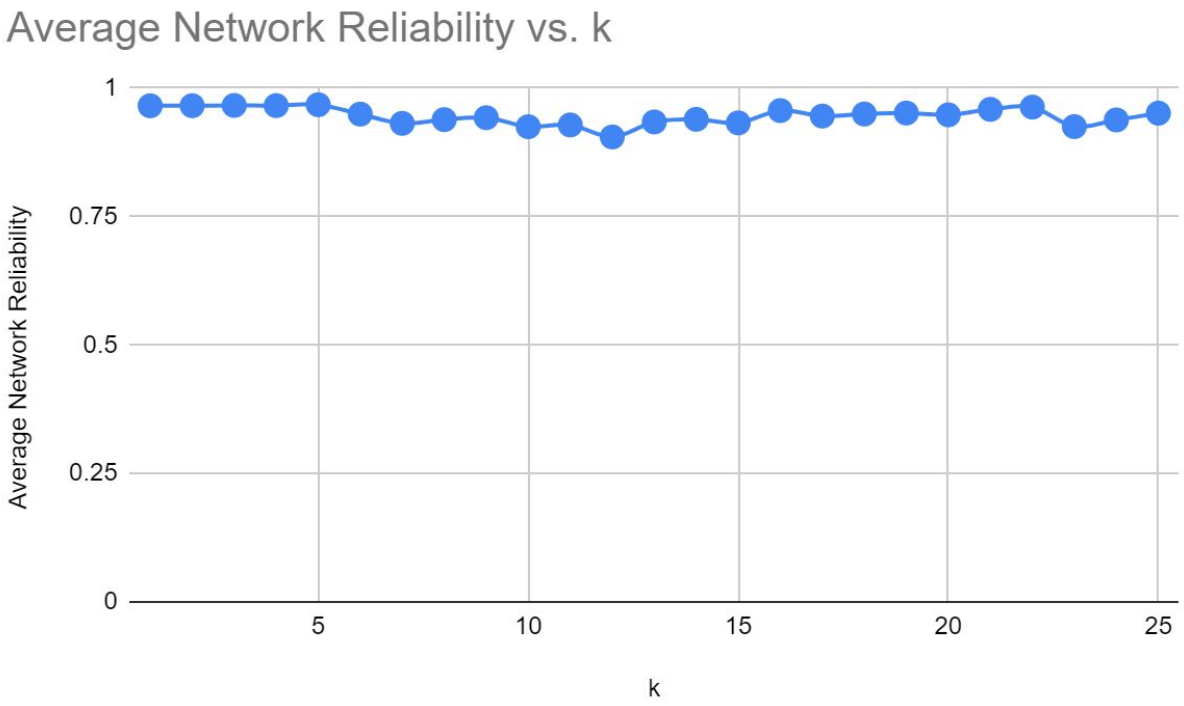
Observation and Analysis:

- Relationship between Network Reliability (R) and Link Reliability(p)



As the value of probability increases, the value of reliability also increases until the value 1 and then it saturates.

- Relationship between Average Network Reliabilty and k:



- Reliability mostly decreases with increase in value of K.
- Probabilistic model for maintaining up down states can be used. The better the probability the better is the reliability.

Reference:

1.wikipedia - About Network Reliability
2.Lecture Notes by Professor Andras Fargo
3.http://seat.massey.ac.nz/

Appendix:

```
import java.text.DecimalFormat;
public class Project3 {
static double sum = 0;
static double rowProb = -1;
static int flag = 0;;

public static int[] calUpDown(int[][] adjMatrix) // To check if the nodes are up or down //
{
int[] temp = new int[1024];
int counter = 0;
for (int i = 0; i < 1024; i++) {
temp[i] = 0;
}
for (int i = 0; i < 1024; i++) {
int[] checkmatrix = new int[5];
for (int k = 0; k < 5; k++) {
checkmatrix[k] = -1;
}
if (adjMatrix[i][0] == 1) {
checkmatrix[0] = 0;
checkmatrix[2] = 2;
}
if (adjMatrix[i][1] == 1) {
checkmatrix[0] = 0;
checkmatrix[1] = 1;
}
if (adjMatrix[i][2] == 1) {
checkmatrix[1] = 1;
checkmatrix[2] = 2;
}
if (adjMatrix[i][3] == 1) {
checkmatrix[1] = 1;
checkmatrix[4] = 4;
}
if (adjMatrix[i][4] == 1) {
checkmatrix[2] = 2;
checkmatrix[3] = 3;
```

```
}
if (adjMatrix[i][5] == 1) {
checkmatrix[2] = 2;
checkmatrix[4] = 4;
}
if (adjMatrix[i][6] == 1){
checkmatrix[3] = 3;
checkmatrix[1] = 1;
}
if (adjMatrix[i][7] == 1) {
checkmatrix[3] = 3;
checkmatrix[2] = 2; }
if (adjMatrix[i][8] == 1) {
checkmatrix[3] = 3;
checkmatrix[4] = 4;
}
if (adjMatrix[i][9] == 1) {
checkmatrix[4] = 4;
checkmatrix[1] = 1;
}
int count = 0;
for (int j = 0; j < 5; j++) {
count = count + checkmatrix[j];
}
if (count == 10) {
temp[counter] = i;
counter++;
}
}
return temp;
}
public static int[][] SetLinks(int[][] adjMatrix) {
for (int i = 0; i < 1024; i++) {

String currentString = String.format("%010d",
Integer.parseInt(Integer.toBinaryString(i)));
for (int j = 0; j < 10; j++) {
if (currentString.charAt(j) == '0') {
adjMatrix[i][j] = 0;
} else { adjMatrix[i][j] = 1;
}
}
}
```

```
return adjMatrix;
}

public static double round(double value, int places) {
if (places < 0) {
throw new IllegalArgumentException();
}
long factor = (long) Math.pow(10, places);
value = value * factor;
long tmp = Math.round(value);
return (double) tmp / factor;
}
public static void main(String[] args) {
int k, i;
int[] randomGenArray = new int[100];
int[][] adjMatrix = new int[1024][10];
int[][] newAdjMatrix = new int[1024][10];
int[] array = new int[1000];
DecimalFormat df = new DecimalFormat("###.##");
for (i = 0; i < 100; i++) {
randomGenArray[i] = -1;
}
sum = 0;
float q = (float) 1.0;
float p = (float) 0.04;
// interval from 0.04 to 1 with 0.04 increments //
if (flag == 0) {
adjMatrix = SetLinks(adjMatrix);
}
array = calUpDown(adjMatrix);
if (flag != 0) {
p = (float) 0.87;
q = (float) 0.87;
}
System.out.println("P\t\t\t\t\t\t\t\t\tReliability");

double d[] = {2.0,0.0,2.0,1.0,4.0,8.0,6.0,7.0,3.0,5.0};//UTD_ID//
for (double id = p; id <= q;) {
for (int j = 0; j < 1000; j++) {
if (array[j] != 0) {
for (int ik = 0; ik < 10; ik++) {
double new_id = Math.pow(id, Math.ceil(d[ik]/3.0));
```

```
if (adjMatrix[array[j]][ik] == 1) {
if (rowProb == -1) {
rowProb = new_id;
} else {
rowProb = rowProb * new_id;
}
} else {
if (rowProb == -1) {
rowProb = (1 - new_id);
} else {
rowProb = rowProb * (1 - new_id);
}
}
}
sum = sum + rowProb;
rowProb = -1;
}
}
System.out.println(id + "    "+ "\t\t\t\t\t" + sum);
id += 0.04;
id = Double.valueOf(df.format(id));
sum = 0;
}
newAdjMatrix = adjMatrix;

flag = 1;
System.out.println("\nValue of p is 0.87\n");
System.out.println("K\t\t\t\t\tReliability");
for (k = 1; k <= 25; k++) {
for (i = 0; i < k; i++)
randomGenArray[i] = (int) (Math.random() * 1024);
int j = 0;
for (i = 1; i <=25; i++); System.out.println(" ");
for (i = 1; i <=25; i++) {
if (randomGenArray[i] != -1) {
for (j = 0; j < 10; j++) {
if (adjMatrix[randomGenArray[i]][j] == 0) {
adjMatrix[randomGenArray[i]][j] = 1;
} else {
adjMatrix[randomGenArray[i]][j] = 0;
}
}
}
```

```
}
sum = 0;
q = (float) 1.0 * 1;
p = (float) 0.0 * 1;
if (flag == 0) {
adjMatrix = SetLinks(adjMatrix);
}
array = calUpDown(adjMatrix);
if (flag != 0) {
p = (float) 0.87 * 1;
q = (float) 0.87 * 1;
}
for (double di = p; di <= q;) {
for (j = 0; j < 1000; j++) {
if (array[j] != 0) {
for (int ki = 0; ki < 10; ki++) {
double new_id = Math.pow(di, Math.ceil(d[ki]/3.0));
if (adjMatrix[array[j]][ki] == 1) {
if (rowProb == -1) {
rowProb = new_id;
} else {
rowProb = rowProb * new_id;
}
} else {
if (rowProb == -1) {
rowProb = (1 - new_id);
} else {
rowProb = rowProb * (1 - new_id);
}
}
}
sum = sum + rowProb;
rowProb = -1;
}
}
System.out.println(k + "\t\t\t\t\t\t" + sum);
di+=0.02;
di = Double.valueOf(df.format(di));
sum = 0;
}
adjMatrix = newAdjMatrix;
for (i = 0; i < k; i++)
randomGenArray[i] = -1;
```

```
}
}
}
```